# Taiwan Credit card default


Group 6

Noah Olsen    Jianing Wang    Jia Chen

# CONTENT

01

# Data Introduction

# Data

**Resource**

The UCI Machine Learning Repository

https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

# Data

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| ID | 30000.0 | 15000.500000 | 8660.398374 | 1.0 | 7500.75 | 15000.5 | 22500.25 | 30000.0 |
| LIMIT_BAL | 30000.0 | 167484.322667 | 129747.661567 | 10000.0 | 50000.00 | 140000.0 | 240000.00 | 1000000.0 |
| SEX | 30000.0 | 1.603733 | 0.489129 | 1.0 | 1.00 | 2.0 | 2.00 | 2.0 |
| EDUCATION | 30000.0 | 1.853133 | 0.790349 | 0.0 | 1.00 | 2.0 | 2.00 | 6.0 |
| MARRIAGE | 30000.0 | 1.551867 | 0.521970 | 0.0 | 1.00 | 2.0 | 2.00 | 3.0 |
| AGE | 30000.0 | 35.485500 | 9.217904 | 21.0 | 28.00 | 34.0 | 41.00 | 79.0 |
| PAY_0 | 30000.0 | -0.016700 | 1.123802 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| PAY_2 | 30000.0 | -0.133767 | 1.197186 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| PAY_3 | 30000.0 | -0.166200 | 1.196868 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| PAY_4 | 30000.0 | -0.220667 | 1.169139 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| PAY_5 | 30000.0 | -0.266200 | 1.133187 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| PAY_6 | 30000.0 | -0.291100 | 1.149988 | -2.0 | -1.00 | 0.0 | 0.00 | 8.0 |
| BILL_AMT1 | 30000.0 | 51223.330900 | 73635.860576 | -165580.0 | 3558.75 | 22381.5 | 67091.00 | 964511.0 |
| BILL_AMT2 | 30000.0 | 49179.075167 | 71173.768783 | -69777.0 | 2984.75 | 21200.0 | 64006.25 | 983931.0 |
| BILL_AMT3 | 30000.0 | 47013.154800 | 69349.387427 | -157264.0 | 2666.25 | 20088.5 | 60164.75 | 1664089.0 |
| BILL_AMT4 | 30000.0 | 43262.948967 | 64332.856134 | -170000.0 | 2326.75 | 19052.0 | 54506.00 | 891586.0 |
| BILL_AMT5 | 30000.0 | 40311.400967 | 60797.155770 | -81334.0 | 1763.00 | 18104.5 | 50190.50 | 927171.0 |
| BILL_AMT6 | 30000.0 | 38871.760400 | 59554.107537 | -339603.0 | 1256.00 | 17071.0 | 49198.25 | 961664.0 |
| PAY_AMT1 | 30000.0 | 5663.580500 | 16563.280354 | 0.0 | 1000.00 | 2100.0 | 5006.00 | 873552.0 |
| PAY_AMT2 | 30000.0 | 5921.163500 | 23040.870402 | 0.0 | 833.00 | 2009.0 | 5000.00 | 1684259.0 |
| PAY_AMT3 | 30000.0 | 5225.681500 | 17606.961470 | 0.0 | 390.00 | 1800.0 | 4505.00 | 896040.0 |
| PAY_AMT4 | 30000.0 | 4826.076867 | 15666.159744 | 0.0 | 296.00 | 1500.0 | 4013.25 | 621000.0 |
| PAY_AMT5 | 30000.0 | 4799.387633 | 15278.305679 | 0.0 | 252.50 | 1500.0 | 4031.50 | 426529.0 |
| PAY_AMT6 | 30000.0 | 5215.502567 | 17777.465775 | 0.0 | 117.75 | 1500.0 | 4000.00 | 528666.0 |
| default payment next month | 30000.0 | 0.221200 | 0.415062 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |

30,000 observations

23 explanatory variables

1 binary target variable [(whether the person defaulted on the next month's payment (1 = Yes, 0 = No) ]

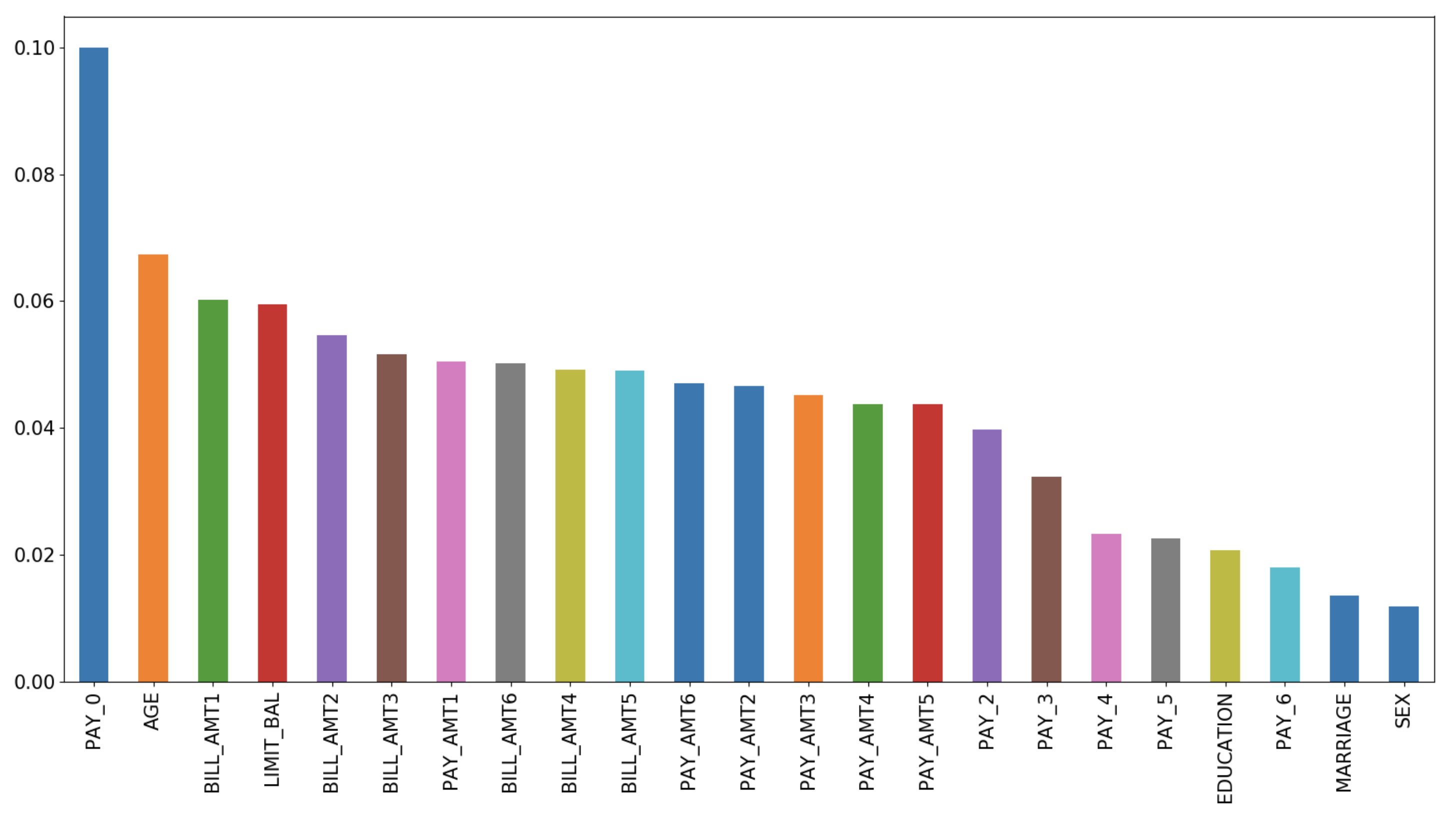
# 02

## Random forest

# Experimental setup

- For the Random Forest model we started by using sklearn RandomForestClassifier function.

- We  started by looking at the feature importance which yielded the plot below.



# Experimental setup

•Given the fact that the dataset doesn’t have that many features to begin with combined with the fact that many of the features appear to have similar importance we decided to use all of the features in the final model.

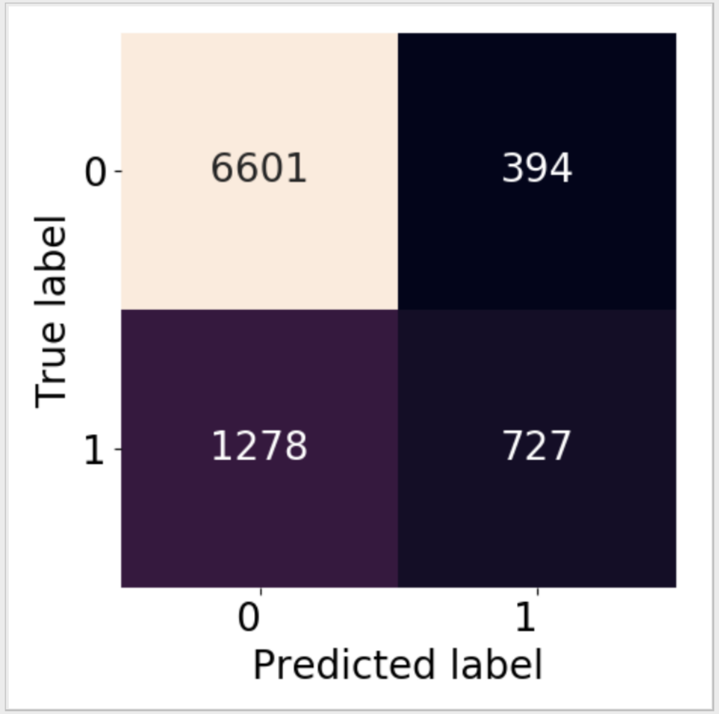
•We  decided to use 100 classifiers as it performed the best without being too computationally intensive. We also tried using both a gini and entropy models.

•Other than that there are not too many tunable parameters in random forest models.

# Results

- The model had an 81.4% accuracy but the confusion matrix shows that the accuracy isn’t equal.
- While the mode is quite good at predicting when someone won’t default on their credit card payment (84% accuracy), the model isn’t very good at predicting when someone will default (64% accuracy).
- Being as the goal is to predict when someone WILL default this might not necessarily be the most appropriate model even though the model sees decent overall performance.

```
Results Using All Features:

Classification Report:
             precision    recall  f1-score   support

          0       0.84      0.94      0.89      6995
          1       0.65      0.36      0.47      2005

avg / total       0.80      0.81      0.79      9000


('Accuracy : ', '81.37777777777778')


('ROC_AUC : ', '76.17347981012445')
```



# 03

## Neuron network

# Experimental setup

- After having experimented with various experimental Neural Network setups

- Using GridSearch with Keras and Stochastic Gradient Descent Algorithms we settled on using three hidden layers with 10 neurons in each layer

- We used a max iteration value of 1000 before giving up on convergence of the model.

- Before running the model we normalized and standardized the dataset.

# Results

- The results from my best performing neural network are shown in the figure below.

- Similarly to the Random Forest Model, the Neural Network Model performed with about 80% accuracy at 79%.

- But also similarly to the Random Forest Model, the Neural Network was much better at predicting when someone won't default on their payment (84%) than when they will default (65%).

- Seeing as this model doesn't perform quite as well as the Random Forest Model it is still preferable to go with the Random Forest Model over this Neural Network Model.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.94 | 0.89 | 6995 |
| 1 | 0.65 | 0.36 | 0.46 | 2005 |
| avg / total | 0.79 | 0.81 | 0.79 | 9000 |

04

# Naïve Bayes

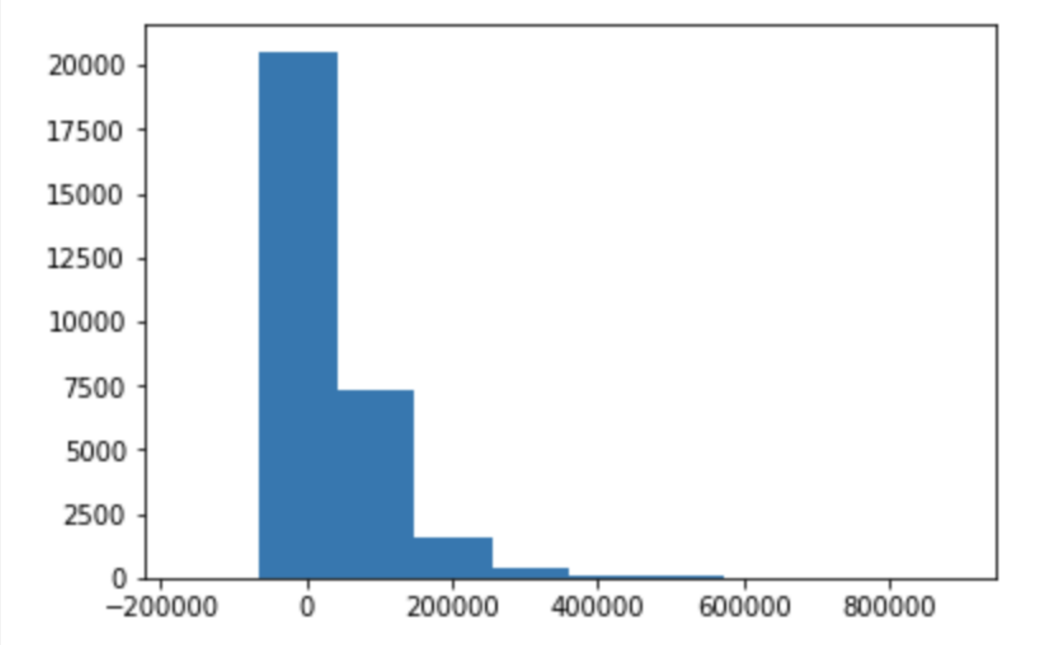
# Naïve Bayes

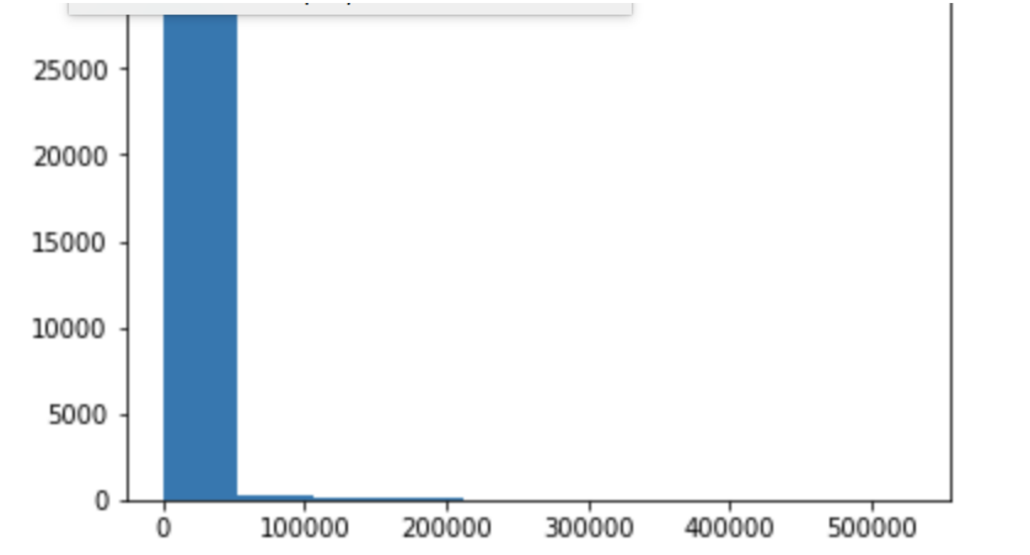
| | LIMIT_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY_0 | PAY_2 | PAY_3 | PAY_4 | PAY_5 | PAY_6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 20000 | 2 | 2 | 1 | 24 | 4.0 | 4.0 | 1.0 | 1.0 | 0.0 | 0.0 |
| **1** | 120000 | 2 | 2 | 2 | 26 | 1.0 | 4.0 | 2.0 | 2.0 | 2.0 | 4.0 |
| **2** | 90000 | 2 | 2 | 2 | 34 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| **3** | 50000 | 2 | 2 | 1 | 37 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| **4** | 50000 | 1 | 2 | 1 | 57 | 1.0 | 2.0 | 1.0 | 2.0 | 2.0 | 2.0 |

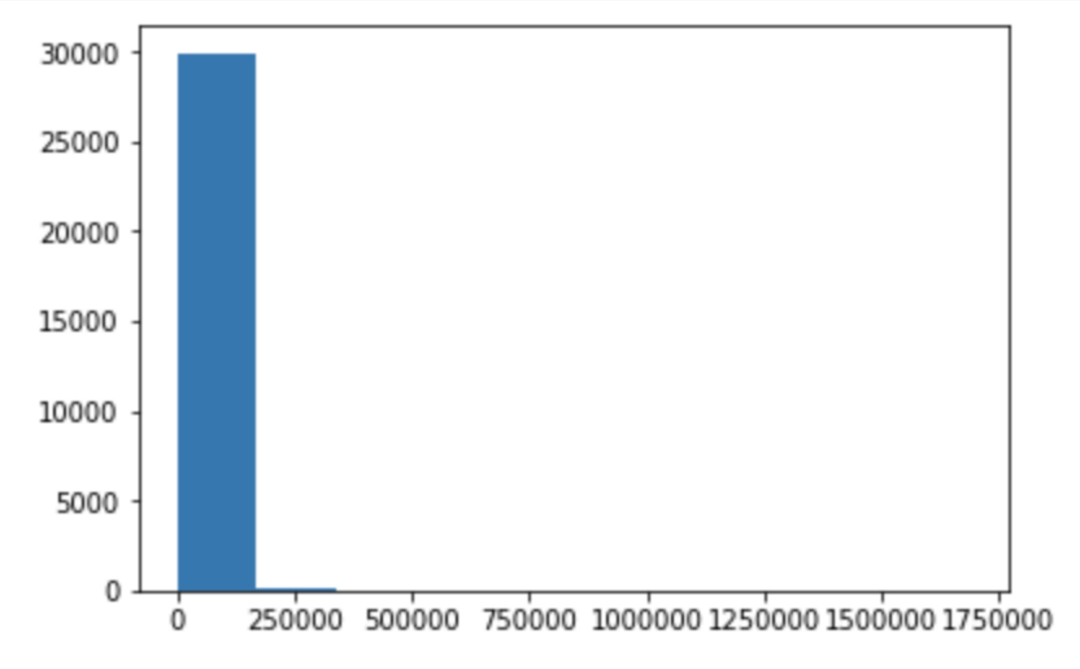
| BILL_AMT1 | BILL_AMT2 | BILL_AMT3 | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 | PAY_AMT1 | PAY_AMT2 | PAY_AMT3 | PAY_AMT4 | PAY_AMT5 | PAY_AMT6 | default payment next month |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3913 | 3102 | 689 | 0 | 0 | 0 | 0 | 689 | 0 | 0 | 0 | 0 | 1 |
| 2682 | 1725 | 2682 | 3272 | 3455 | 3261 | 0 | 1000 | 1000 | 1000 | 0 | 2000 | 1 |
| 29239 | 14027 | 13559 | 14331 | 14948 | 15549 | 1518 | 1500 | 1000 | 1000 | 1000 | 5000 | 0 |
| 46990 | 48233 | 49291 | 28314 | 28959 | 29547 | 2000 | 2019 | 1200 | 1100 | 1069 | 1000 | 0 |
| 8617 | 5670 | 35835 | 20940 | 19146 | 19131 | 2000 | 36681 | 10000 | 9000 | 689 | 679 | 0 |

# Naïve Bayes

```
def distribution(i):
    plt.hist(d.iloc[:,i])
    plt.show()
distribution(2)
```



# Naïve Bayes

## Condition for Multinomial function

1. Discrete variables
2. Non-negative data

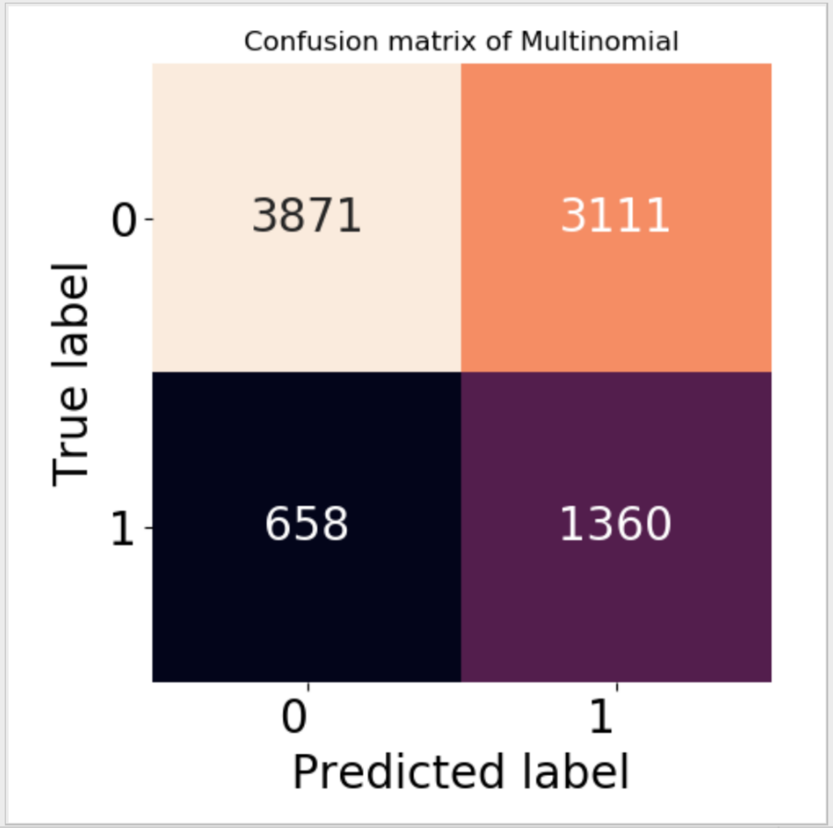
```
#switch the continuous variables to discrete variables and non-negtitive datas
for j in range(5, 11):
    d.iloc[:,j] = pd.cut(d.iloc[:, j],11 ,labels=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
    d.iloc[:, j] = d.iloc[:, j].astype('float64')

for i in range(11, 23):
    d.iloc[:, i] = pd.cut(d.iloc[:, i], 3,labels=[0, 1, 2])
    d.iloc[:, i] = d.iloc[:, i].astype('float64')
```

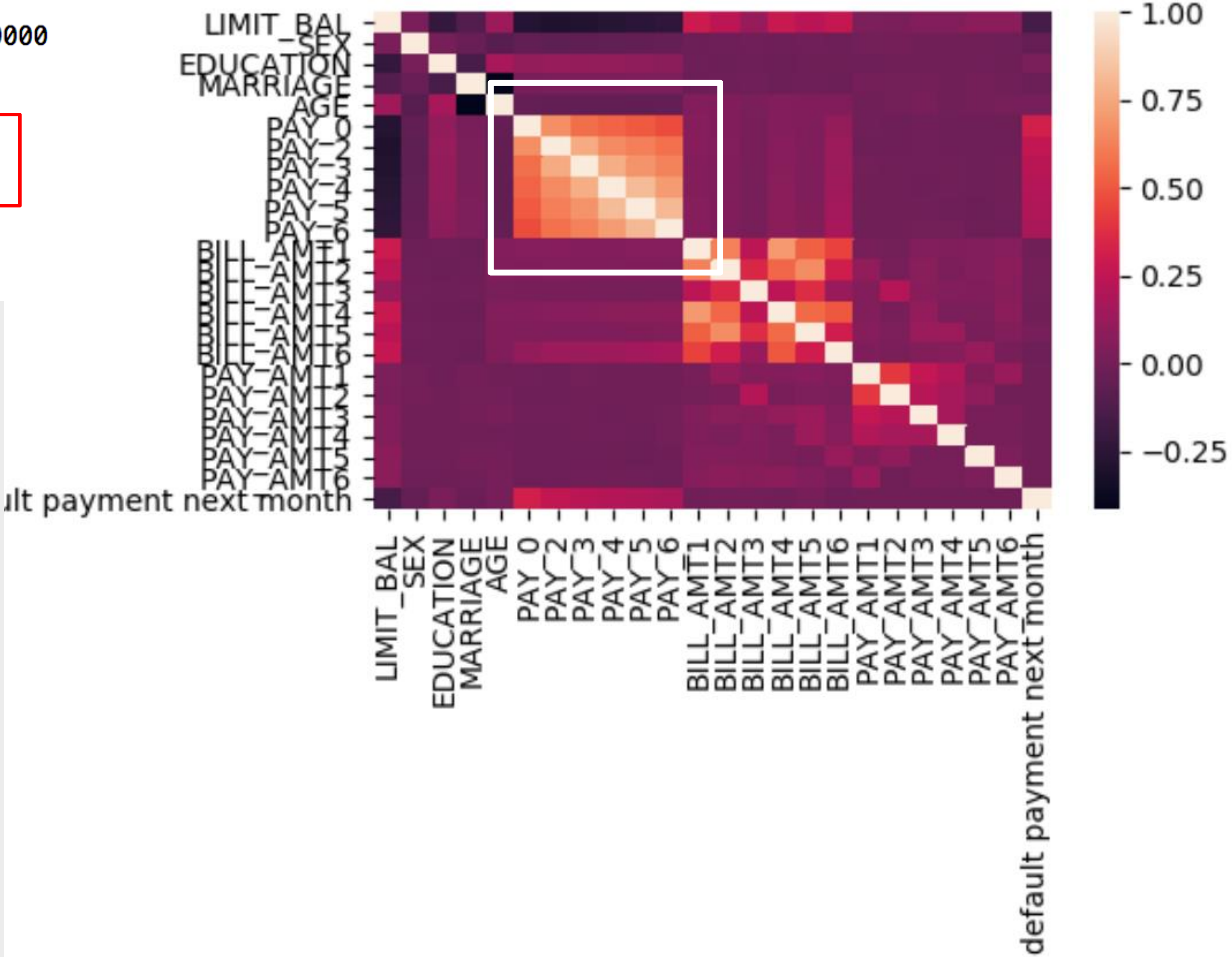
**Accuracy : 58%**

```
Classification Report of Multinomial_2 :
              precision    recall  f1-score   support

        0.0       0.85      0.55      0.67      6982
        1.0       0.30      0.67      0.42      2018

avg / total       0.73      0.58      0.62      9000
```

Accuracy of Multinomial_2 : $\boxed{58.12222222222222}$

ROC_AUC of Multinomial_2 :  66.29088206144698



# Naïve Bayes

## Gaussian – Normalization

```
#normalizing the data
d_gaussian=pd.read_csv('cc_default_data.csv', sep=',', header=0)
d_gaussian.drop(["ID"], axis=1, inplace=True)
predata=d_gaussian.values[:,(11,12,13,14,15,16,17,18,19,20,21,22)]

scaledata=preprocessing.scale(predata)
scaledata.shape
x1=d_gaussian.values[:,:11]
x=np.c_[x1,scaledata]
```

Decrease the probability of overfitting.

Each feature has same scale to the outcome which is another assumption in Naïve Bayes

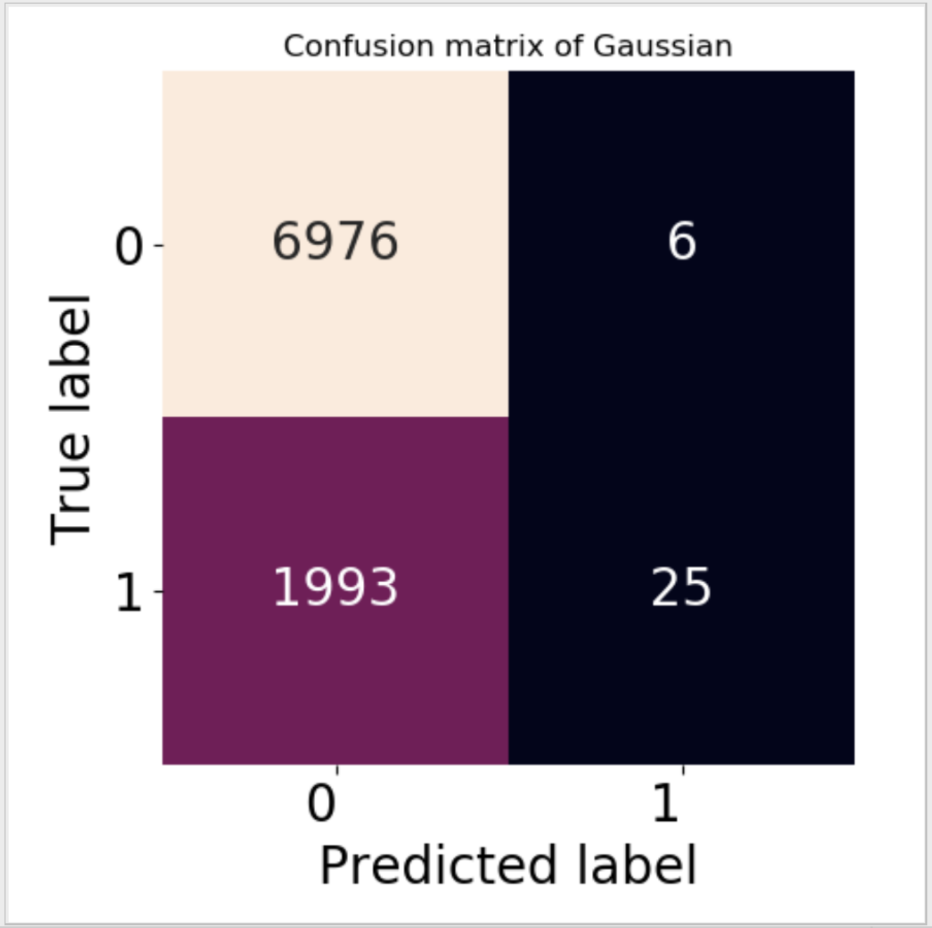
# Naïve Bayes

Classification Report of Gaussian:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 1.00 | 0.87 | 6982 |
| 1 | 0.81 | 0.01 | 0.02 | 2018 |
| avg / total | 0.78 | 0.78 | 0.68 | 9000 |

Accuracy of Gaussian : 77.7888888888889

ROC_AUC of Gaussian: 68.9915651715483




The result rises up to 77.8%.

# Naïve Bayes

The accuracy of Multinomial function is 58.1%.

The accuracy in Gaussian function is higher than that, but the accuracy for class I was only 1%, which means this model is not able to predict the default payment.

It may because the accuracy of Naïve Bayes can be reduced if applied to large amounts of data or if the correlations among features are strong.

Also, the types of the feature are various, it is hard to train if we only use the algorithm of Naïve Bayes before a bunch of complex data preparation work done.

Therefore, Naïve Bayes is not an appropriate method to apply in this dataset.

05

# SVM

# SVM

## Data Pre-processing

- Data cleaning drops unnecessary columns such as “ID”
- Types: continuous variables and discrete variables (categorical variables)
  - ➢ continuous variables --- MinMaxScaler of sklearn.preprocessing

```
# normalize continuous columns such as LimitBalance, BillAmount and PaymentAmount
min_max_data = preprocessing.MinMaxScaler()
X_Part1_minmax = DataFrame(min_max_data.fit_transform(data.values[:, :1]))
X_Part2_minmax = DataFrame(min_max_data.fit_transform(data.values[:, 11:23]))
X_Part5_minmax = DataFrame(min_max_data.fit_transform(data.values[:, 4:5]))
```

    - ✓ MaxAbsScaler, scale, normalize

  - ➢ categorical features such as marital status --- pandas.get_dummies

```
# encoding the categorical features such as gender, marital status
X_Part3 = pd.get_dummies(data.iloc[:, 1:4])
X_Part4 = pd.get_dummies(data.iloc[:, 5:11])
```

# SVM

## The Choice of Kernel

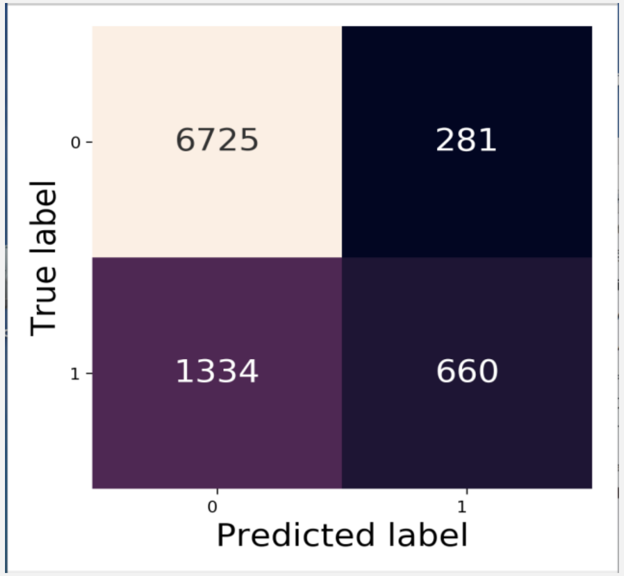
- It is recommended that try a RBF kernel first, because it usually works well. However, it is good to try the other types of kernels if you have enough time to do so (Alexandre Kowalczyk, 2017)
- RBF  --- 80%
- Linear --- 80%
- RBF and Linear worked equally well ---- Linear kernel

# SVM

## Result

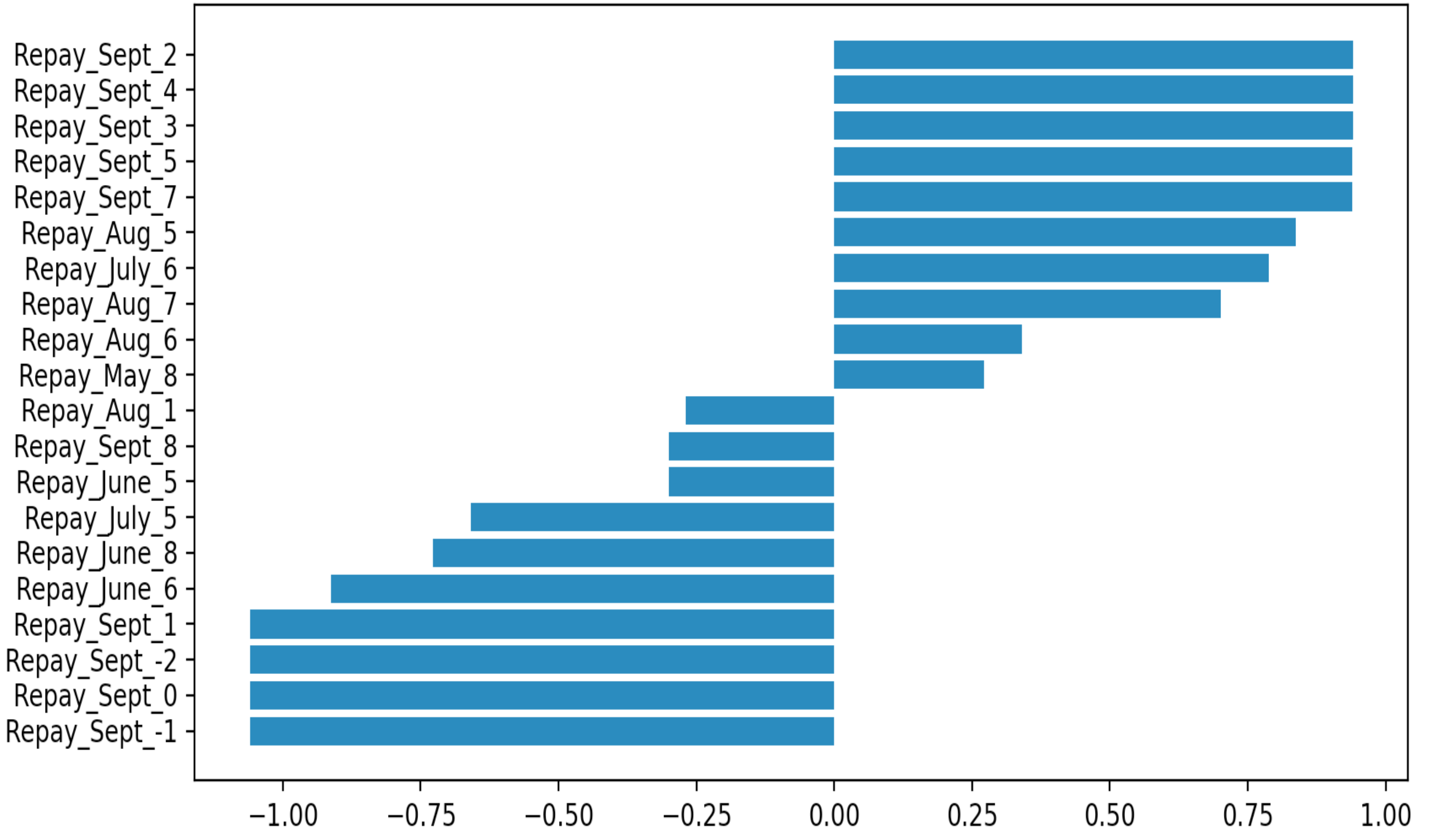
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.96 | 0.89 | 7006 |
| 1 | 0.70 | 0.33 | 0.45 | 1994 |
| avg / total | 0.80 | 0.82 | 0.79 | 9000 |




- The accuracy of SVM model for this data is 80%.
- However, the precision of predicting class 1 is lower than class 0. It means that this model is stronger when predicting a client who won't default payment to the class who won't default. In contrast, it is weak to detect the client who will default.

# SVM




- Repay_Sept_2:
- It means client delay payment for 2 months in September.

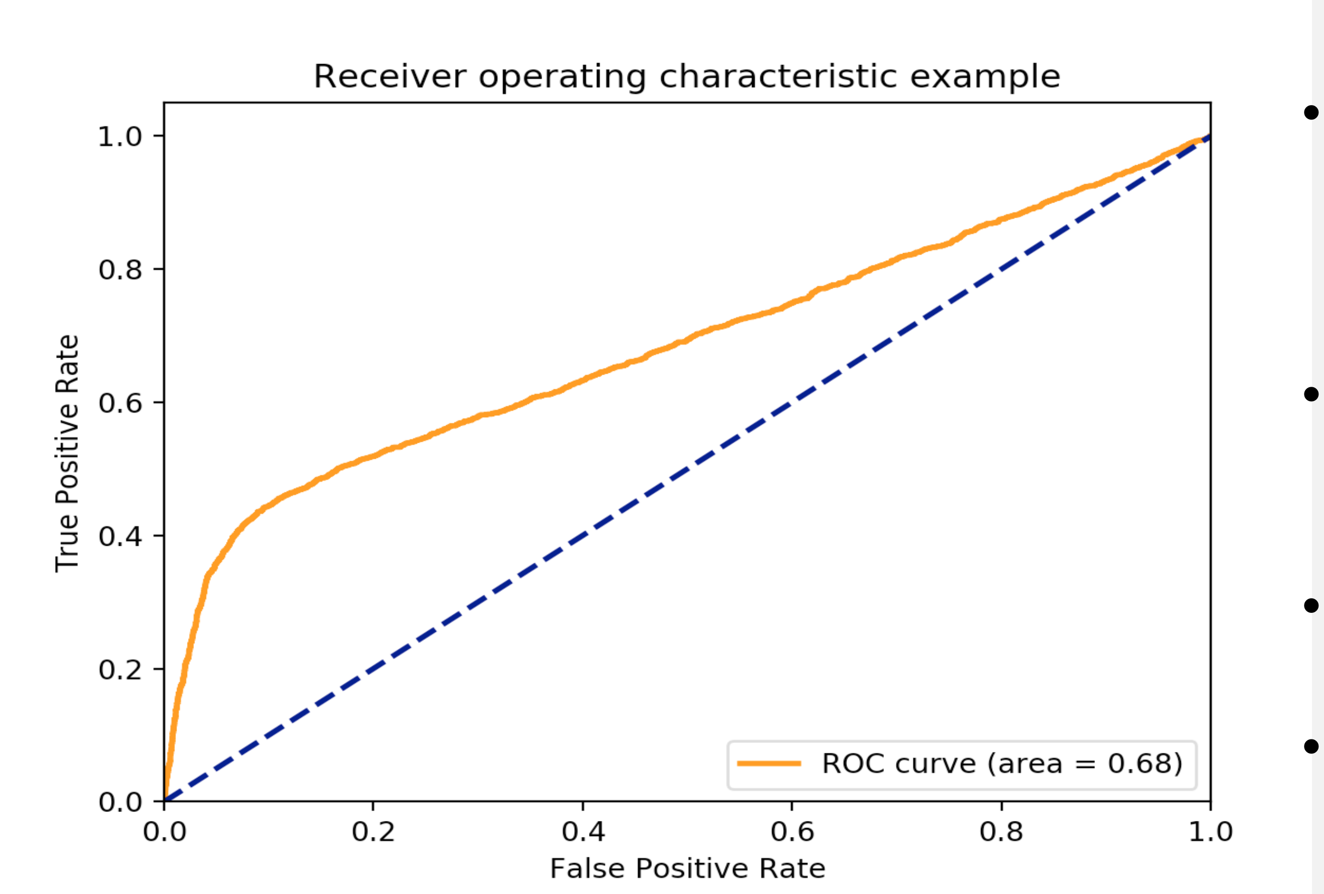
- -1 = pay duly;
- 1 = payment delay for one month;
- 2 = payment delay for two months;

…

9 = payment delay for two months and above.

- Repayment history is predominantly important.
- Repayment status in September !!!

# SVM




- The figure shows the ROC curve and area. As can be seen, the ROC curve is in the above of blue dash line.

- It illustrates that this model is better than random classifier.

- However, it also demonstrates the fact that the accuracy of this model is notperfect.
- 80%.

# 06

# Conclusion

- In terms of accuracy:

  - RF (81.4%)
  - SVM (80.0%)
  - NN (79.0%)
  - Multinomial NB (58.1%)
  - Gaussian NB (77.8%)

- Precision of class 0 ( no default) and class 1 (default):

  - SVM 70.0%
  - RF 65.0%)
  - NN (65.0%)
  - Gaussian NB(81%, recalled 1%)

- Above all, RF and SVM classifiers are good choices for this dataset.
- In contrast, NB is not an appropriate method for this dataset.
- For all of the models, the precision of predicting default needs to be enhanced.

# Thank You


Group 6
August, 2018